Φροντιστήριο 7 – Σκελετοί Λύσεων

1. Χρησιμοποιούμε τη δομή

```
typedef struct Node{
        int key;
        struct Node *left;
        struct Node *right;
} node;
```

και υποθέτουμε πως ένα δυαδικό δένδρο αναζήτησης είναι υλοποιημένο ως δείκτης στη ρίζα του δένδρου, δηλαδή, έχει τύπο *node.
Το πρόβλημα λύνεται με την πιο κάτω αναδρομική διαδικασία.

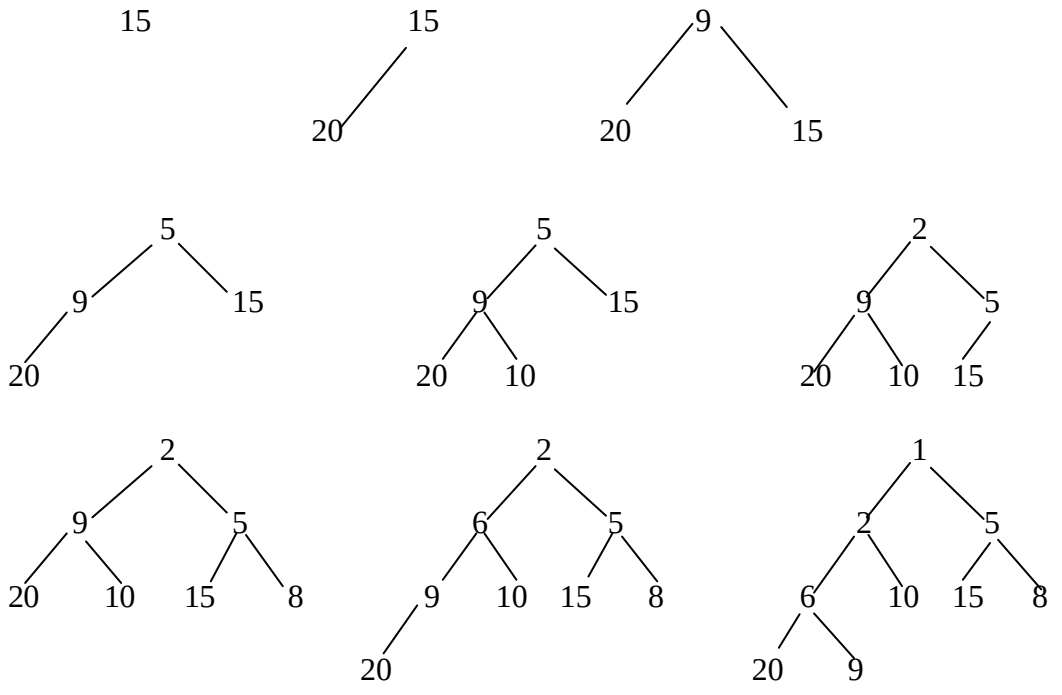
```
Between(node *p, int k1, intk2){
        if (p == NULL)
             return;

        if (p->key > k2)
             Between(p->left, k1, k2);

        if (p->key < k1)
             Between(p->right, k1, k2);

        if (k1 < p->key < k2) {
             Between(p->right, k1, k2);
             print p->key;
             Between(p->left, k1, k2);
        }
}
```

2. (α) Ξεκινώντας με ένα άδειο σωρό, εισάγουμε διαδοχικά τα στοιχεία: 15, 20, 9, 5, 10, 2, 8, 6, και 1.



3. Υποθέτουμε ότι ο σωρός είναι υλοποιημένος με τη δομή:

```
struct heap{
      int array[];
      int maxsize;
      int size;
}
```

Η ζητούμενη διαδικασία έχει ως εξής:

```
Level (int k, heap *A){
      position = 1;
      level = 1;
      while (level < k AND position < A->size) {
            position = 2*position;
            level++;
      }
      if  (level == k) {
            maxpos = 2*position-1;
            while ( position <= min(maxpos, A->size)) {
                  print A->array[position];
                  position++;
            }
      }
}
```

Ο χρόνος εκτέλεσης της διαδικασίας είναι της τάξης O(k + m) όπου m είναι ο αριθμός των στοιχείων του επιπέδου k. Στη χείριστη περίπτωση όπου ζητείται εκτύπωση του τελευταίου επιπέδου ενός σωρού με n στοιχεία, $k \in O(\lg n)$ και $m \in O(n)$, επομένως η διαδικασία έχει χρόνο εκτέλεσης $O(\lg n + n)$.